Filter Type VII Design

Take Home Endsem Submission

EE338: Digital Signal Processing

Report

Full Name	Roll No.
Prajwal Nayak	22B4246

Instructor: Prof. Vikram Gadre

Department of Electrical Engineering
Indian Institute of Technology Bombay
Powai, Mumbai, India

April 9, 2025

Contents

1 | Introduction

We are given a M value that gives most of the specifications of the filter. The Band Specifications: All frequencies are in kiloHertz (kHz). There are two groups of frequency bands, which will be used in specifying the filters ahead. For each group of frequency bands, we pass the argument, which is an integer ranging from 0 to 10. The frequency band in each group is specified according to the argument.
Group I of Frequency Bands: The frequency band in this group is (40 + 5D) to (70 + 5D)
Group II of Frequency Bands: The frequency band in this group is (170 + 5D) to (200 + 5D).

$$M = 11Q + R$$

The frequency band from Group I can be obtained by passing the argument D = Q and the frequency band from Group II can be obtained by passing the argument D = R.
For given $\mathbf{M = 114}$, we get $\mathbf{Q = 10}$ and $\mathbf{R = 4}$.
An analog signal is bandlimited to 280 kHz. It is ideally sampled, with a sampling rate of 630 kHz. We wish to build a series of discrete time filters, as described below, to extract specific frequency bands of an analog signal or to suppress specific frequency bands of the signal.
(i) For all filters, the passband AND stopband tolerances are 0.15 in magnitude. That is, the filter magnitude response (note: NOT magnitude squared) must lie between 1.15 and 0.85 in the passband; and between 0 and 0.15 in the stopband. For the IIR Filter, the passband magnitude response must lie between 1 and 0.85.
(ii) For bandpass filters, the transition bands are 5 kHz on either side of each passband. For bandstop filters, the transition bands are 5 kHz on either side of each stopband.

Group I frequency band range is from 90 to 120
Group II frequency band range is from 190 to 220
Transition band for Group I frequency band is 85 to 90 and 120 to 125
Transition band for Group II frequency band is 185 to 190 and 220 to 225

2 | Filter Design Specifications

Given that $\delta_1 = 0.15$ and $\delta_2 = 0.15$. We have multiple passbands and stopbands:

- $\Omega_{pI1} = 90$
- $\Omega_{pI2} = 120$
- $\Omega_{pII1} = 190$
- $\Omega_{pII2} = 220$
- $\Omega_{sI1} = 85$
- $\Omega_{sI2} = 125$
- $\Omega_{sII1} = 185$
- $\Omega_{sII2} = 225$

The Filter Type VII is an FIR MultiBandPass Filter. I am using two Bandpass filters in parallel.

$$H_{\text{final}}(e^{j\omega}) = H_{\text{BandPass1}}(e^{j\omega}) + H_{\text{BandPass2}}(e^{j\omega})$$

$$h_{\text{final}}[n] = h_{\text{BandPass1}}[n] + h_{\text{BandPass2}}[n]$$

2.1 | Individual Filter Calculation

We are now using two filters in parallel, and we want our final filter to satisfy the tolerance requirement. I want to use the same tolerance δ for both the bandpass filters. Since the filters have passbands far enough, I have decided to use 0.15 as tolerance for each bandpass filter and then check with the results if the filters satisfy the specifications.

3 | Procedure

3.1 | Frequency Table

We first list the frequencies which form the passband and the stopband edges. We then convert it to normalised frequency to make it independent of the sampling rate.

$$\omega = \frac{2\pi f}{f_s}$$

The table for the bands is given below:

Category	Un-normalised Frequency (kHz) [f]	Normalised Frequency (rad/s) [ω]
Stopband for Group 1	0-85	0.0-0.8477
Passband for Group 1	90-120	0.8976-1.1968
Stopband Intermediate	125-185	1.2467-1.8451
Passband for Group 2	190-220	1.8949-2.1941
Stopband for Group 2	225-315	2.2440-3.1416

Table 3.1: Frequency Table with Normalized Frequencies

3.2 | Target Filter

The filter for which we aim is the following:

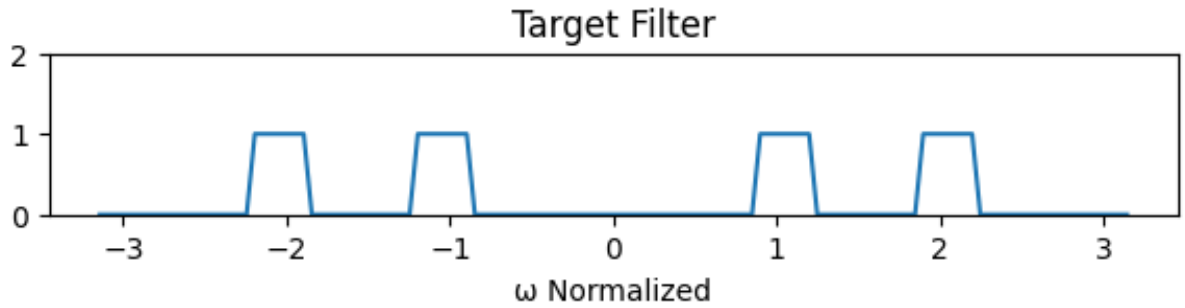


Figure 3.1: Target Filter

We achieve this filter by having two parallel bandpass filters.

1. To achieve a multi-bandpass filter, the task can be divided into designing two separate bandpass filters.
2. The final multi-bandpass filter can then be obtained by cascading or implementing a parallel combination of the two filters. Since the
3. My approach :Two Bandpass Filters in parallel
4. Since the two passbands are far away, the tolerance for the individual bandpass filters need not be changed as the stopband value would be extremely small far away from the stop band.
5. So for the individual bandpass filters we use tolerance = 0.15.
6. We implement the above target filter using parallel of two bandpass filters, we name them BP1 and BP2.

$$H_{\text{final}}(e^{j\omega}) = H_{\text{BP1}}(e^{j\omega}) + H_{\text{BP2}}(e^{j\omega})$$

$$h_{\text{final}}[n] = h_{\text{BP1}}[n] + h_{\text{BP2}}[n]$$

3.3 | Designing BandPass Filter BPF1

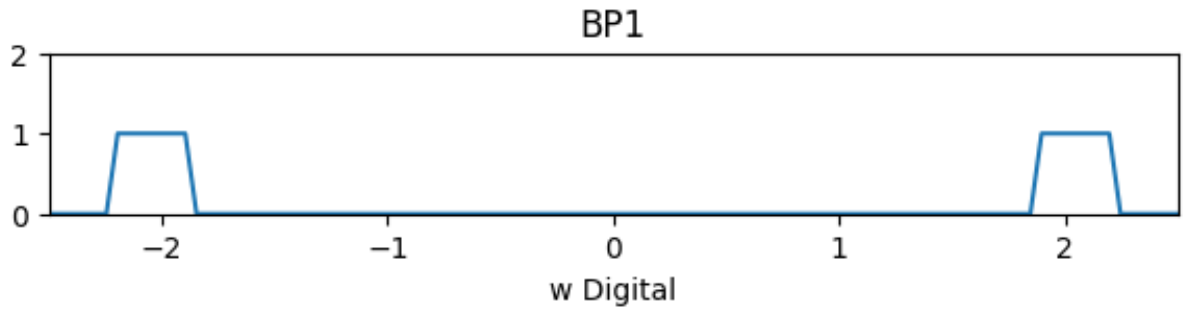


Figure 3.2: BandPass Filter BPF1 Target

1. We first design two lowpassfilters : LP11 and LP12. This is done because we are given only the kaiser window coefficients obtaining method for a lowpass filter.
2. We subtract them to get the bandpass filter
3. We have to make sure that the tolerances are within limits. We use tolerance $= 0.15/2 = 0.075$ for each low pass filter.

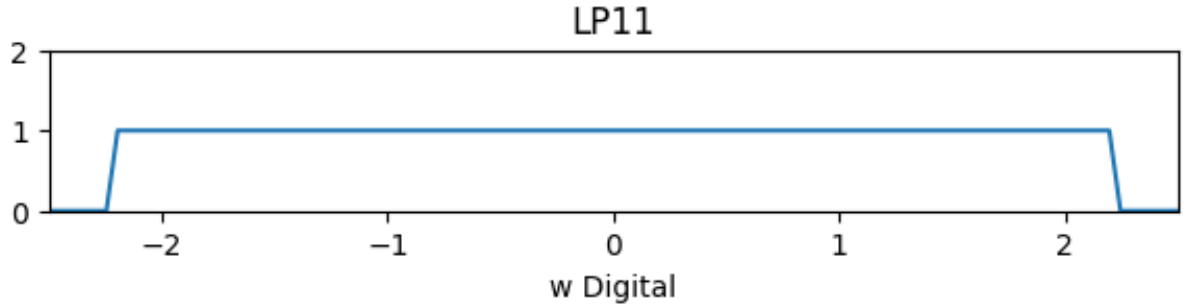


Figure 3.3: Low Pass Filter LP11 corresponding to BPF1

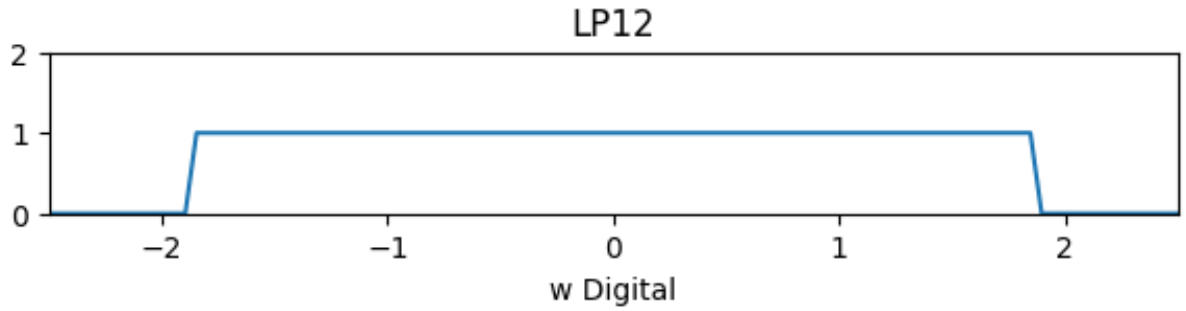


Figure 3.4: Low Pass Filter LP12 corresponding to BPF1

3.3.1 | Procedure for Designing the individual LowPass Filters

We need a LowPass Filter satisfying our constraints but having a FINITE IMPULSE RESPONSE. So the infinite impulse response of an ideal lowpass filter $lp11_{ideal}[n]$ will have to be truncated or slightly modified by multiplying by a window function (Kaiser window as required by the problem statement) $kaiser[n]$ to satisfy the constraints as well as behave like a Low pass filter $lp11[n]$.

$$lp11[n] = lp11_{ideal}[n] \times kaiser[n]$$

3.3.2 | Kaiser Window

$$Kaiser[n] = \begin{cases} \frac{I_0\left(\beta N\sqrt{1-\left(\frac{n}{N}\right)^2}\right)}{I_0(\beta N)} = \frac{I_0\left(\alpha\sqrt{1-\left(\frac{n}{N}\right)^2}\right)}{I_0(\alpha)} & \beta > 0, \alpha = N\beta \quad \text{if } -N \leq n \leq N \\ 0, & \text{otherwise} \end{cases}$$

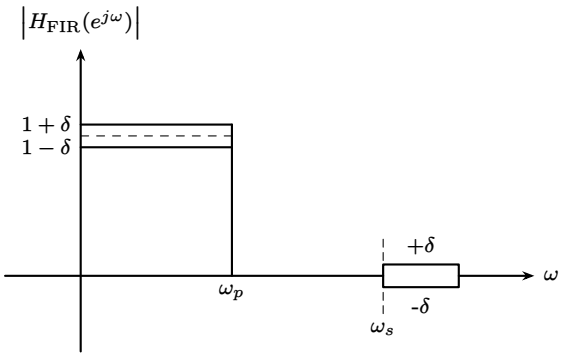
where

$$I_0(x) = \text{modified Bessel function of first kind and order 0 in x} \triangleq 1 + \sum_{\ell=1}^{\infty} \left(\frac{x}{2}\right)^{2\ell} \Big/ (\ell!)^2$$

This is how I manually coded the Bessel Function and the Kaiser window.

```
def Bessel_function(x, terms=25):
    s = 1
    l = 1
    while l <= terms:
        s = s + (((x/2)**l) / (math.factorial(l)))**2
        l = l + 1
    return s
def kaiser_window(alpha, N):
    value = np.zeros(2*N + 1)
    for k in range(-N, N+1):
        numerator = Bessel_function(alpha * math.sqrt(1 - (k/N)**2))
        denominator = Bessel_function(alpha)
        value[k + N] = numerator / denominator
    return value
```

3.3.3 | Kaiser Window Empirical Design Equations for LowPass Filters



To design a filter as shown above using kaiser window, we need to tune the parameters as given:

- $\Delta\omega_T = \omega_s - \omega_p$
- $A = -20\log_{10}(\delta)$
- $\alpha = \beta N$
- Choose N according to $(2N+1) \geq 1 + \frac{A-8}{2.285\,\Delta\omega_T}$.
- Now choose α (and hence β) according to:

$$\alpha = \begin{cases} 0.1102\,(A-8.7), & \text{for } A > 50, \\ 0.5842\,(A-21)^{0.4} + 0.07886\,(A-21), & \text{for } 21 \leq A \leq 50, \\ 0, & \text{for } A < 21. \end{cases}$$

- Note that these values are empirical values and might go out of tolerance limits. The values will have to be tweaked around to get a perfect fit as per given constraints.

3.3.4 | Kaiser Window Parameters for Implementation of Low Pass Filter

The following code has been used to get the parameter values ($\Delta\omega_T$, A, α and N) for the kaiser window.

```
tol = 0.075
delta_w_t = w3[18] - w3[17] #this is constant for all the low pass filters (transition
    width)
A = -20*math.log10(tol)
print("The value of delta_w_t is,",delta_w_t)
print("The value of A is,",A)
N=math.ceil((A-8)/(2*2.285*delta_w_t))#this is th eminimum N value we should choose
print("The minimum value of N is :",N)
# Let us take a slightly higher value of N
N = N + 2
print("We use the value of N to be :",N)
alpha = 0
if(A>=21 and A<=50):
  alpha= 0.5842*((A-21)**0.4) + 0.07886*(A-21)
elif(A>50):
  alpha = 0.1102*(A-8.7)
print("Alpha value given by the equation is: ", alpha)
alpha = alpha + 0.069975#let us use a slightly higher value of alpha
print("The tweaked value of alpha is ",alpha)
```

1. The value of $\Delta\omega_T$ is going to be constant for all filters as it is the transition width and it is fixed to be 5kHz, which in normalised frequency is 0.0498.
2. The value of A is calculated using the formula given in the section 3.3.3.
3. The value of N and α is also calculated as given above but since the formulae are empirically given, tweaking was required.
4. After trying by hit and trial, i reached a minimum value of N and α, below which the tolerance limit was being crossed.
5. The output of the above code is as follows:
   - The value of delta_w_t is, 0.049866550056980596
   - The value of A is, 22.498774732165998
   - The minimum value of N is : 64
   - We use the value of N to be : 66
   - Alpha value given by the equation is: 0.8050341948769562
   - The tweaked value of alpha is 0.8750091948769562

For these values, we get the kaiser window as follows:

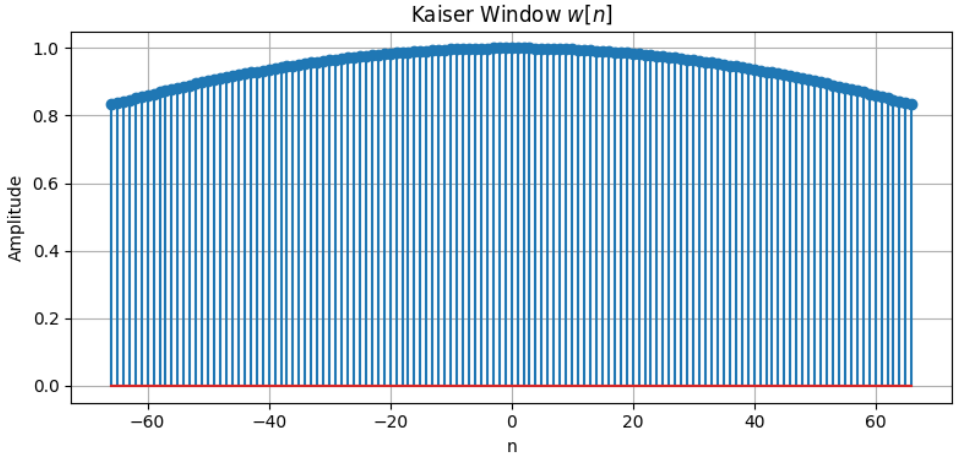


Figure 3.5: Kaiser Window Coefficients

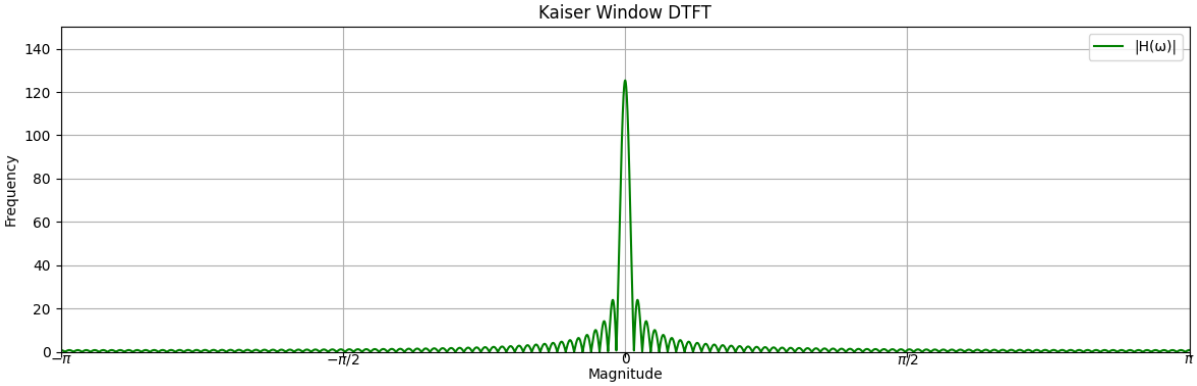


Figure 3.6: Kaiser Window DTFT

3.3.5 | Kaiser Window Coefficients

Kaiser[-66]: 0.8328	Kaiser[-65]: 0.8376	Kaiser[-64]: 0.8423	Kaiser[-63]: 0.8470	Kaiser[-62]: 0.8516
Kaiser[-61]: 0.8562	Kaiser[-60]: 0.8607	Kaiser[-59]: 0.8651	Kaiser[-58]: 0.8695	Kaiser[-57]: 0.8738
Kaiser[-56]: 0.8780	Kaiser[-55]: 0.8822	Kaiser[-54]: 0.8863	Kaiser[-53]: 0.8904	Kaiser[-52]: 0.8944
Kaiser[-51]: 0.8983	Kaiser[-50]: 0.9021	Kaiser[-49]: 0.9059	Kaiser[-48]: 0.9096	Kaiser[-47]: 0.9132
Kaiser[-46]: 0.9168	Kaiser[-45]: 0.9203	Kaiser[-44]: 0.9237	Kaiser[-43]: 0.9271	Kaiser[-42]: 0.9304
Kaiser[-41]: 0.9336	Kaiser[-40]: 0.9368	Kaiser[-39]: 0.9398	Kaiser[-38]: 0.9428	Kaiser[-37]: 0.9457
Kaiser[-36]: 0.9486	Kaiser[-35]: 0.9514	Kaiser[-34]: 0.9541	Kaiser[-33]: 0.9567	Kaiser[-32]: 0.9593
Kaiser[-31]: 0.9618	Kaiser[-30]: 0.9642	Kaiser[-29]: 0.9665	Kaiser[-28]: 0.9687	Kaiser[-27]: 0.9709
Kaiser[-26]: 0.9730	Kaiser[-25]: 0.9750	Kaiser[-24]: 0.9770	Kaiser[-23]: 0.9788	Kaiser[-22]: 0.9806
Kaiser[-21]: 0.9824	Kaiser[-20]: 0.9840	Kaiser[-19]: 0.9855	Kaiser[-18]: 0.9870	Kaiser[-17]: 0.9884
Kaiser[-16]: 0.9897	Kaiser[-15]: 0.9910	Kaiser[-14]: 0.9921	Kaiser[-13]: 0.9932	Kaiser[-12]: 0.9942
Kaiser[-11]: 0.9951	Kaiser[-10]: 0.9960	Kaiser[-9]: 0.9967	Kaiser[-8]: 0.9974	Kaiser[-7]: 0.9980
Kaiser[-6]: 0.9986	Kaiser[-5]: 0.9990	Kaiser[-4]: 0.9994	Kaiser[-3]: 0.9996	Kaiser[-2]: 0.9998
Kaiser[-1]: 1.0000	Kaiser[0]: 1.0000	Kaiser[1]: 1.0000	Kaiser[2]: 0.9998	Kaiser[3]: 0.9996
Kaiser[4]: 0.9994	Kaiser[5]: 0.9990	Kaiser[6]: 0.9986	Kaiser[7]: 0.9980	Kaiser[8]: 0.9974
Kaiser[9]: 0.9967	Kaiser[10]: 0.9960	Kaiser[11]: 0.9951	Kaiser[12]: 0.9942	Kaiser[13]: 0.9932
Kaiser[14]: 0.9921	Kaiser[15]: 0.9910	Kaiser[16]: 0.9897	Kaiser[17]: 0.9884	Kaiser[18]: 0.9870
Kaiser[19]: 0.9855	Kaiser[20]: 0.9840	Kaiser[21]: 0.9824	Kaiser[22]: 0.9806	Kaiser[23]: 0.9788
Kaiser[24]: 0.9770	Kaiser[25]: 0.9750	Kaiser[26]: 0.9730	Kaiser[27]: 0.9709	Kaiser[28]: 0.9687
Kaiser[29]: 0.9665	Kaiser[30]: 0.9642	Kaiser[31]: 0.9618	Kaiser[32]: 0.9593	Kaiser[33]: 0.9567
Kaiser[34]: 0.9541	Kaiser[35]: 0.9514	Kaiser[36]: 0.9486	Kaiser[37]: 0.9457	Kaiser[38]: 0.9428
Kaiser[39]: 0.9398	Kaiser[40]: 0.9368	Kaiser[41]: 0.9336	Kaiser[42]: 0.9304	Kaiser[43]: 0.9271
Kaiser[44]: 0.9237	Kaiser[45]: 0.9203	Kaiser[46]: 0.9168	Kaiser[47]: 0.9132	Kaiser[48]: 0.9096
Kaiser[49]: 0.9059	Kaiser[50]: 0.9021	Kaiser[51]: 0.8983	Kaiser[52]: 0.8944	Kaiser[53]: 0.8904
Kaiser[54]: 0.8863	Kaiser[55]: 0.8822	Kaiser[56]: 0.8780	Kaiser[57]: 0.8738	Kaiser[58]: 0.8695
Kaiser[59]: 0.8651	Kaiser[60]: 0.8607	Kaiser[61]: 0.8562	Kaiser[62]: 0.8516	Kaiser[63]: 0.8470
Kaiser[64]: 0.8423	Kaiser[65]: 0.8376	Kaiser[66]: 0.8328		

Table 3.2: Kaiser WIndow Coefficients

3.3.6 | Implementing the Low Pass Filters LP11 and LP12

This is the code for a general ideal lowpass filter implulse response coefficients truncated to 2N+1 terms.

```
def ideal_lowpass(wc,N):
  value = np.zeros(2*N+1)
  for k in range(-N,N+1):
    if(k!=0):
      numerator = math.sin(wc*k)
      denominator = k*math.pi
      value[k+N] = numerator/denominator
```

```
    else:
      value[k+N] = wc/math.pi
return value
```

1. Here all I did was calculated the IDTFT of an ideal lowpass filter with cutoff frequency ω_c and truncated it to 2N+1 because that is how much we need for this filter design and also that we cant store all the infinite impulses which are the part of the sinc function as we have limited memory.

2. The frequency response of an ideal low-pass filter is given by:

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| \leq \omega_c \\ 0, & \omega_c < |\omega| \leq \pi \end{cases}$$

The inverse Discrete-Time Fourier Transform (IDTFT) is defined as:

$$h[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n}\, d\omega = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n}\, d\omega = \frac{1}{2\pi} \cdot \left[\frac{e^{j\omega n}}{jn} \right]_{-\omega_c}^{\omega_c} = \frac{1}{2\pi} \cdot \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{jn}$$

$$h[n] = \frac{1}{2\pi} \cdot \frac{2j \sin(\omega_c n)}{jn} = \frac{\sin(\omega_c n)}{\pi n}, \quad n \neq 0$$

For $n = 0$, the integral becomes:

$$h[0] = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} 1\, d\omega = \frac{\omega_c}{\pi}$$

Final Result:

$$h[n] = \begin{cases} \frac{\omega_c}{\pi}, & n = 0 \\ \frac{\sin(\omega_c n)}{\pi n}, & n \neq 0 \end{cases}$$

This is the ideal low-pass filter impulse response in the time domain.

3. The final result above is what was implemented in the code above.

4. We multiply these coefficients with the kaiser window coefficients to the the FIR low pass filter which satisfies our tolerance limits. The results I got from my code are attached.Note that I used the value of ω_c for the low pass filters as $\frac{\omega_p + \omega_s}{2}$

5. In a similar fashion we obtained the LowPass Filter L12 as well.

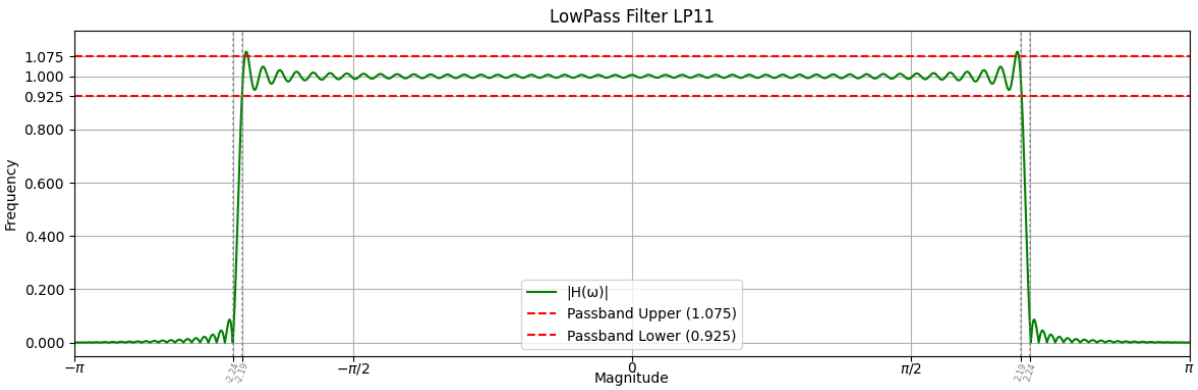


Figure 3.7: Magnitude response of LP11 ($lp11[n] = lp11_ideal[n] \times kaiser[n]\ \forall n$)

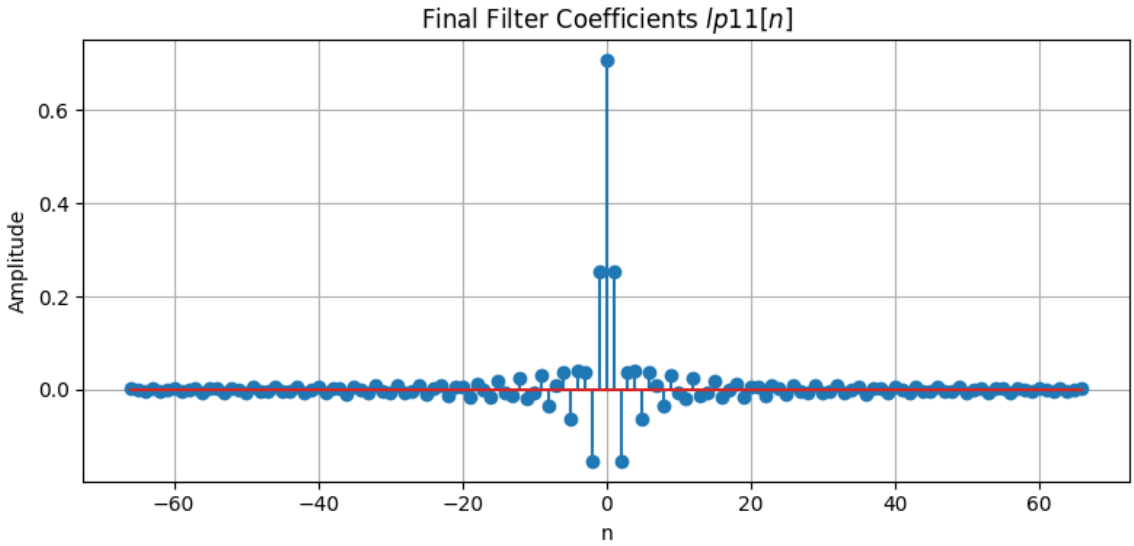


Figure 3.8: Low Pass Filter LP11 Coefficients

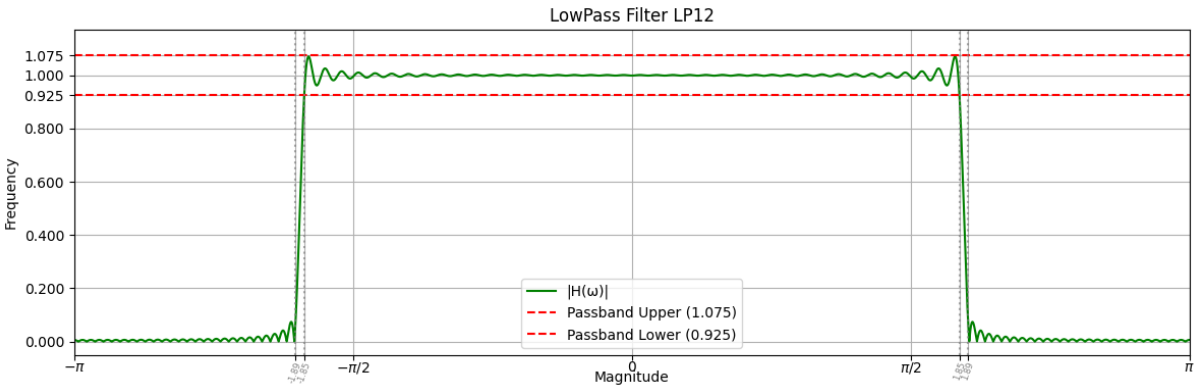


Figure 3.9: Magnitude response of LP12 ($lp12[n] = lp12_ideal[n] \times kaiser[n]\ \forall n$)

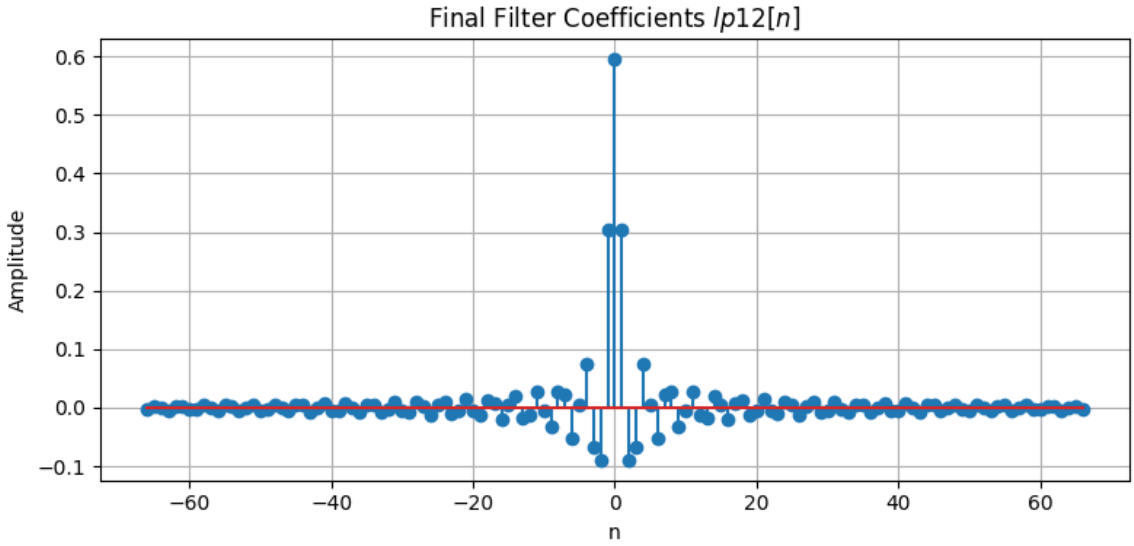


Figure 3.10: Low Pass Filter LP12 Coefficients

3.4 | Implementing the Bandpass Filter BPF1

This was the only line for the Bandpass filter implementation

```
1    bpf1 = lp11 - lp12
```

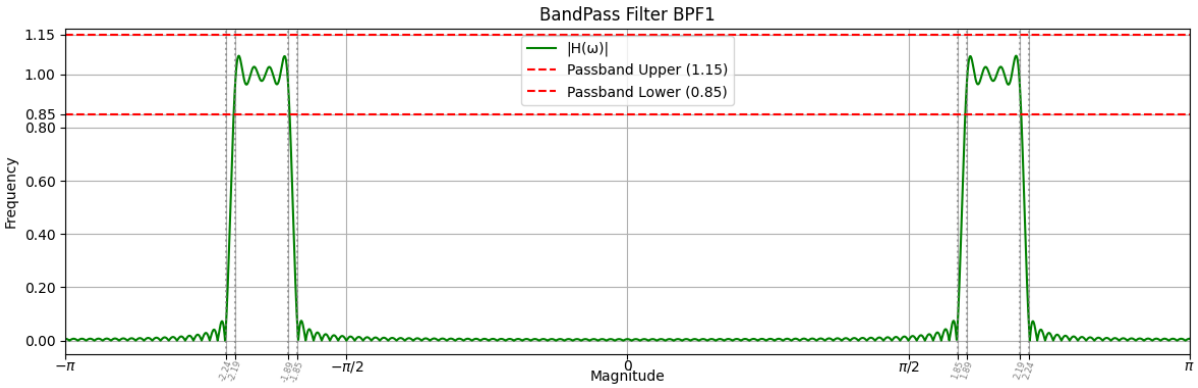


Figure 3.11: Magnitude Response of BandPass Filter

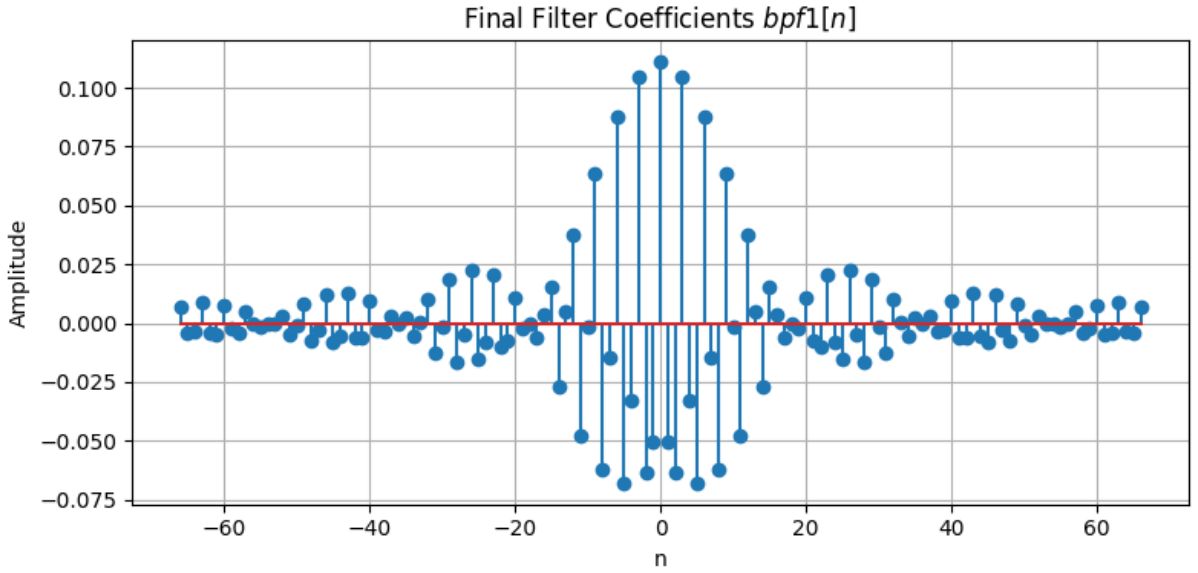


Figure 3.12: BandPass Filter Coefficients

Table 3.3: Band Pass Filter 1 Coefficients

BPF1[-66]: 0.0069	BPF1[-65]: -0.0045	BPF1[-64]: -0.0038	BPF1[-63]: 0.0086	BPF1[-62]: -0.0039
BPF1[-61]: -0.0049	BPF1[-60]: 0.0078	BPF1[-59]: -0.0023	BPF1[-58]: -0.0043	BPF1[-57]: 0.0047
BPF1[-56]: -0.0006	BPF1[-55]: -0.0014	BPF1[-54]: -0.0000	BPF1[-53]: 0.0000	BPF1[-52]: 0.0033
BPF1[-51]: -0.0046	BPF1[-50]: -0.0009	BPF1[-49]: 0.0085	BPF1[-48]: -0.0077	BPF1[-47]: -0.0031
BPF1[-46]: 0.0122	BPF1[-45]: -0.0081	BPF1[-44]: -0.0054	BPF1[-43]: 0.0129	BPF1[-42]: -0.0061
BPF1[-41]: -0.0060	BPF1[-40]: 0.0095	BPF1[-39]: -0.0028	BPF1[-38]: -0.0036	BPF1[-37]: 0.0027
BPF1[-36]: 0.0000	BPF1[-35]: 0.0023	BPF1[-34]: -0.0056	BPF1[-33]: 0.0007	BPF1[-32]: 0.0105
BPF1[-31]: -0.0129	BPF1[-30]: -0.0013	BPF1[-29]: 0.0184	BPF1[-28]: -0.0166	BPF1[-27]: -0.0051
BPF1[-26]: 0.0227	BPF1[-25]: -0.0154	BPF1[-24]: -0.0082	BPF1[-23]: 0.0207	BPF1[-22]: -0.0099
BPF1[-21]: -0.0074	BPF1[-20]: 0.0107	BPF1[-19]: -0.0024	BPF1[-18]: 0.0000	BPF1[-17]: -0.0063
BPF1[-16]: 0.0036	BPF1[-15]: 0.0154	BPF1[-14]: -0.0273	BPF1[-13]: 0.0046	BPF1[-12]: 0.0377
BPF1[-11]: -0.0475	BPF1[-10]: -0.0016	BPF1[-9]: 0.0635	BPF1[-8]: -0.0623	BPF1[-7]: -0.0148
BPF1[-6]: 0.0877	BPF1[-5]: -0.0680	BPF1[-4]: -0.0326	BPF1[-3]: 0.1049	BPF1[-2]: -0.0635
BPF1[-1]: -0.0504	BPF1[0]: 0.1111	BPF1[1]: -0.0504	BPF1[2]: -0.0635	BPF1[3]: 0.1049
BPF1[4]: -0.0326	BPF1[5]: -0.0680	BPF1[6]: 0.0877	BPF1[7]: -0.0148	BPF1[8]: -0.0623
BPF1[9]: 0.0635	BPF1[10]: -0.0016	BPF1[11]: -0.0475	BPF1[12]: 0.0377	BPF1[13]: 0.0046

BPF1[14]: -0.0273	BPF1[15]: 0.0154	BPF1[16]: 0.0036	BPF1[17]: -0.0063	BPF1[18]: 0.0000
BPF1[19]: -0.0024	BPF1[20]: 0.0107	BPF1[21]: -0.0074	BPF1[22]: -0.0099	BPF1[23]: 0.0207
BPF1[24]: -0.0082	BPF1[25]: -0.0154	BPF1[26]: 0.0227	BPF1[27]: -0.0051	BPF1[28]: -0.0166
BPF1[29]: 0.0184	BPF1[30]: -0.0013	BPF1[31]: -0.0129	BPF1[32]: 0.0105	BPF1[33]: 0.0007
BPF1[34]: -0.0056	BPF1[35]: 0.0023	BPF1[36]: 0.0000	BPF1[37]: 0.0027	BPF1[38]: -0.0036
BPF1[39]: -0.0028	BPF1[40]: 0.0095	BPF1[41]: -0.0060	BPF1[42]: -0.0061	BPF1[43]: 0.0129
BPF1[44]: -0.0054	BPF1[45]: -0.0081	BPF1[46]: 0.0122	BPF1[47]: -0.0031	BPF1[48]: -0.0077
BPF1[49]: 0.0085	BPF1[50]: -0.0009	BPF1[51]: -0.0046	BPF1[52]: 0.0033	BPF1[53]: 0.0000
BPF1[54]: -0.0000	BPF1[55]: -0.0014	BPF1[56]: -0.0006	BPF1[57]: 0.0047	BPF1[58]: -0.0043
BPF1[59]: -0.0023	BPF1[60]: 0.0078	BPF1[61]: -0.0049	BPF1[62]: -0.0039	BPF1[63]: 0.0086
BPF1[64]: -0.0038	BPF1[65]: -0.0045	BPF1[66]: 0.0069		

3.5 | Designing and Implementing BandPass Filter BPF2

The theory is exactly the same as previously for BPF1.

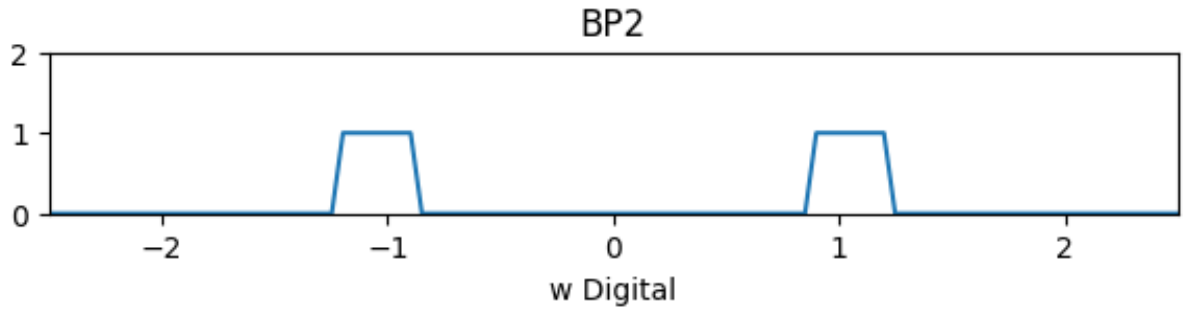


Figure 3.13: BandPass Filter 2 Target

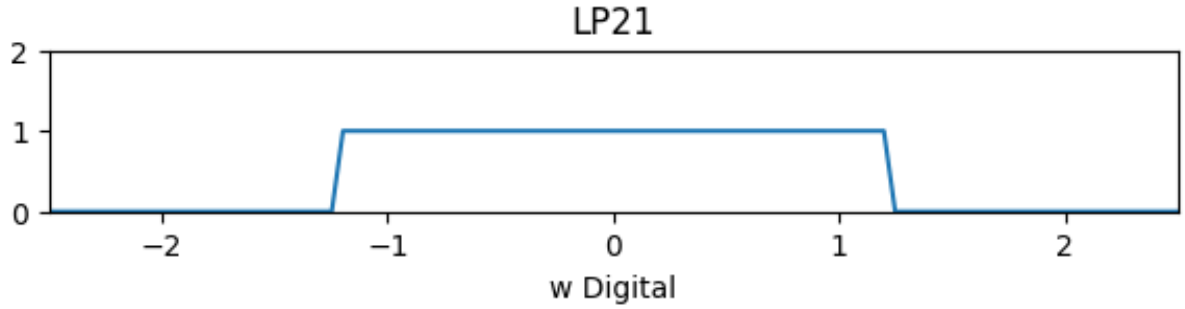


Figure 3.14: Low Pass Filter LP21 corresponding to BPF2

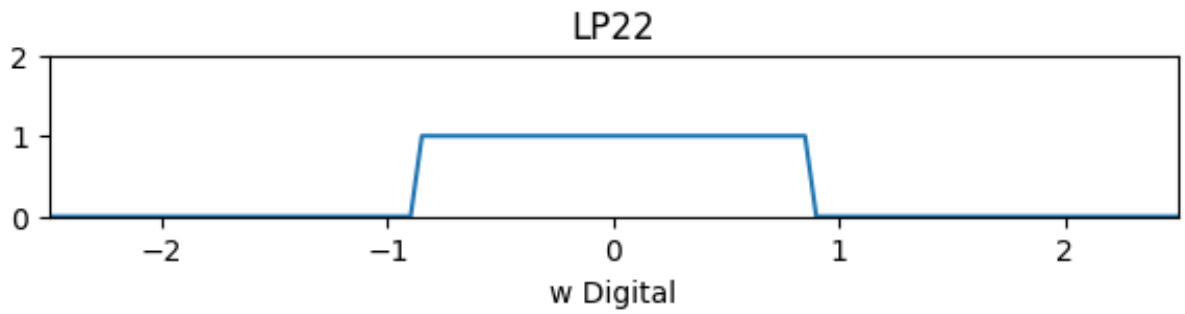


Figure 3.15: Low Pass Filter LP22 corresponding to BPF2

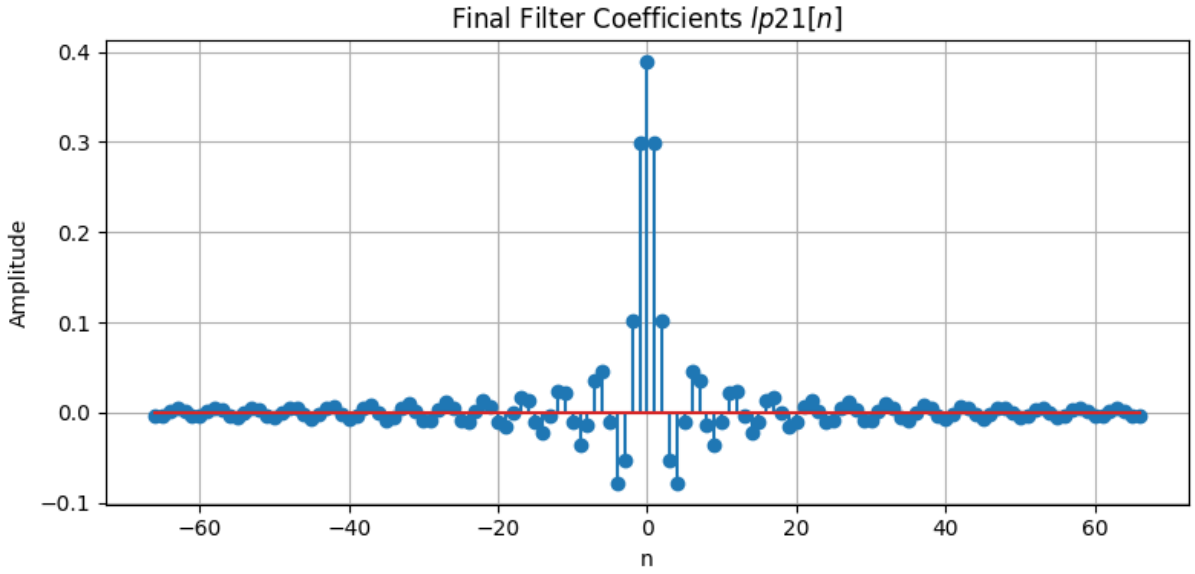


Figure 3.16: Low Pass Filter LP21 Coefficients

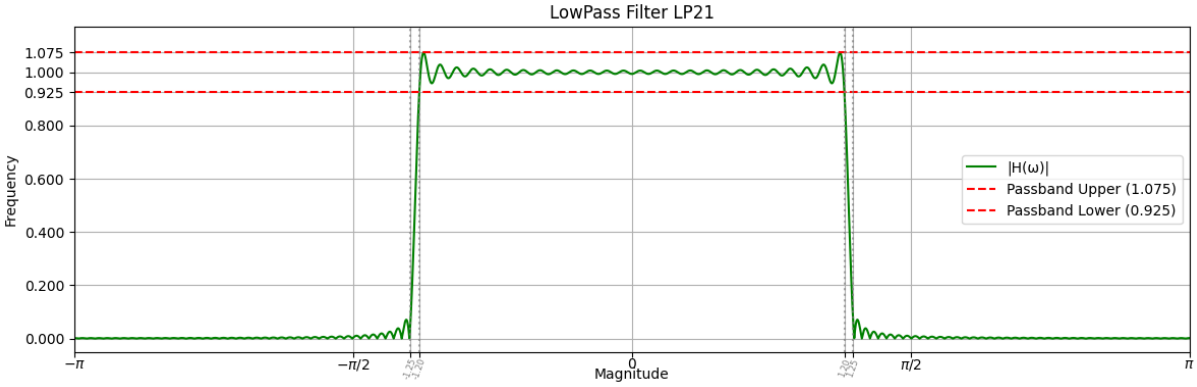


Figure 3.17: Magnitude response of LP21 ($lp21[n] = lp21_ideal[n] \times kaiser[n]\ \forall n$)

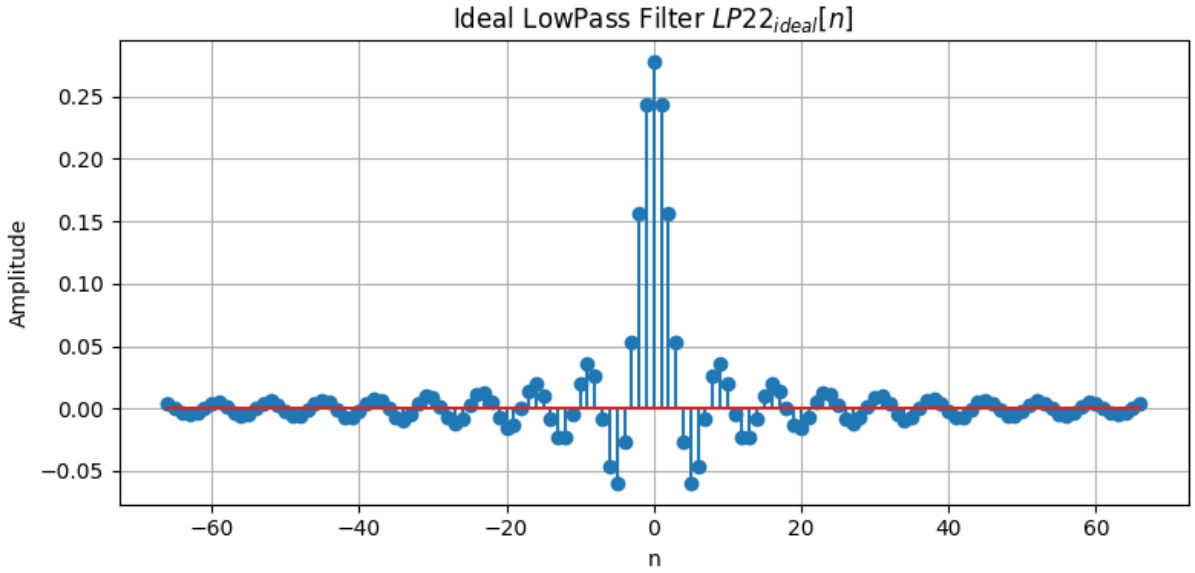


Figure 3.18: Low Pass Filter LP22 Coefficients

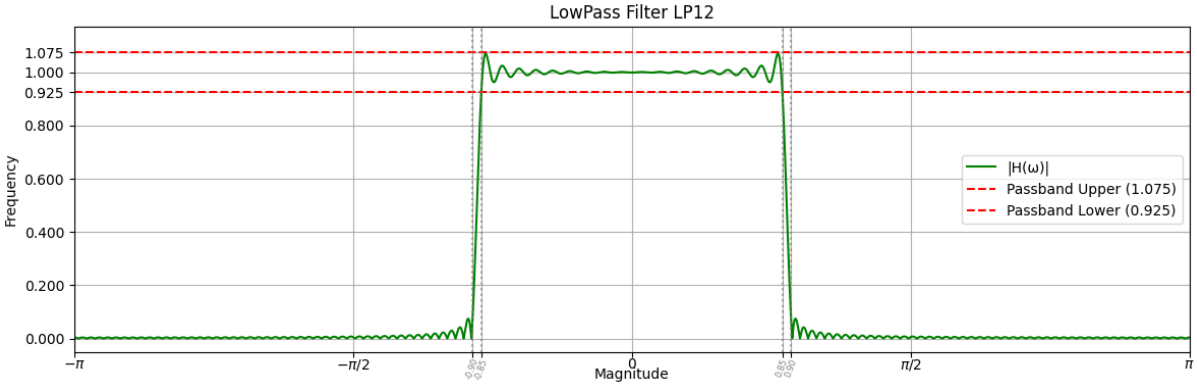


Figure 3.19: Magnitude response of LP22 ($lp22[n] = lp22_ideal[n] \times kaiser[n]\ \forall n$)

3.5.1 | Implementing the BandpassFilter BPF2

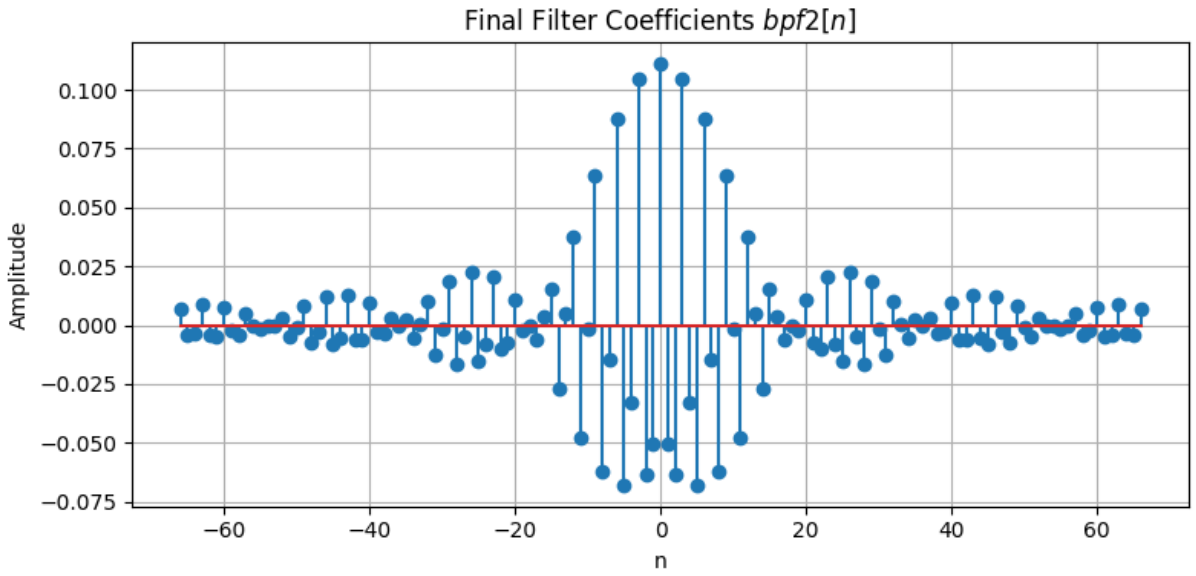


Figure 3.20: BandPass Filter Coefficients

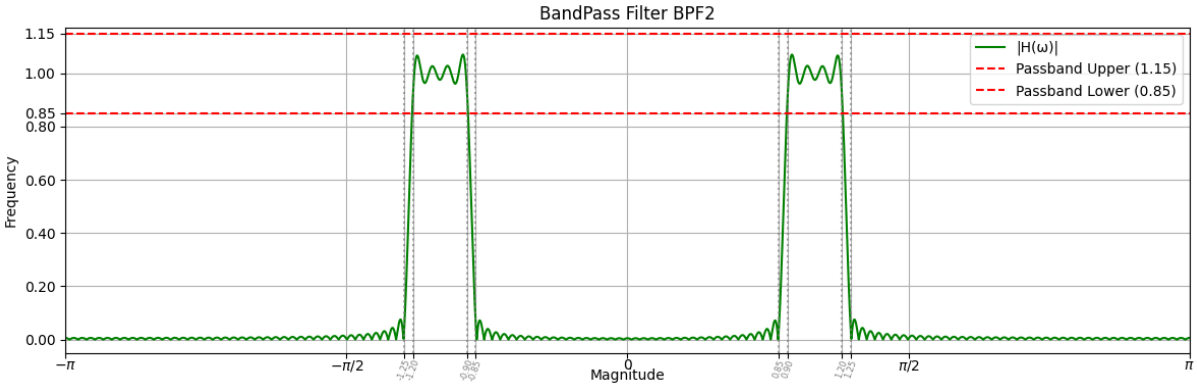


Figure 3.21: Magnitude Response of BandPass Filter

Table 3.4: Band Pass Filter 2 Coefficients

BPF2[-66]: -0.0070	BPF2[-65]: -0.0039	BPF2[-64]: 0.0041	BPF2[-63]: 0.0086	BPF2[-62]: 0.0043
BPF2[-61]: -0.0042	BPF2[-60]: -0.0079	BPF2[-59]: -0.0036	BPF2[-58]: 0.0031	BPF2[-57]: 0.0049
BPF2[-56]: 0.0017	BPF2[-55]: -0.0009	BPF2[-54]: 0.0000	BPF2[-53]: 0.0009	BPF2[-52]: -0.0019
BPF2[-51]: -0.0056	BPF2[-50]: -0.0037	BPF2[-49]: 0.0045	BPF2[-48]: 0.0104	BPF2[-47]: 0.0058
BPF2[-46]: -0.0062	BPF2[-45]: -0.0130	BPF2[-44]: -0.0066	BPF2[-43]: 0.0064	BPF2[-42]: 0.0122
BPF2[-41]: 0.0056	BPF2[-40]: -0.0048	BPF2[-39]: -0.0077	BPF2[-38]: -0.0027	BPF2[-37]: 0.0014
BPF2[-36]: -0.0000	BPF2[-35]: -0.0015	BPF2[-34]: 0.0031	BPF2[-33]: 0.0092	BPF2[-32]: 0.0061
BPF2[-31]: -0.0076	BPF2[-30]: -0.0177	BPF2[-29]: -0.0100	BPF2[-28]: 0.0108	BPF2[-27]: 0.0229
BPF2[-26]: 0.0117	BPF2[-25]: -0.0117	BPF2[-24]: -0.0224	BPF2[-23]: -0.0104	BPF2[-22]: 0.0091
BPF2[-21]: 0.0149	BPF2[-20]: 0.0054	BPF2[-19]: -0.0029	BPF2[-18]: 0.0000	BPF2[-17]: 0.0032
BPF2[-16]: -0.0067	BPF2[-15]: -0.0210	BPF2[-14]: -0.0145	BPF2[-13]: 0.0186	BPF2[-12]: 0.0457
BPF2[-11]: 0.0271	BPF2[-10]: -0.0312	BPF2[-9]: -0.0705	BPF2[-8]: -0.0391	BPF2[-7]: 0.0426
BPF2[-6]: 0.0918	BPF2[-5]: 0.0487	BPF2[-4]: -0.0511	BPF2[-3]: -0.1061	BPF2[-2]: -0.0544
BPF2[-1]: 0.0553	BPF2[0]: 0.1111	BPF2[1]: 0.0553	BPF2[2]: -0.0544	BPF2[3]: -0.1061
BPF2[4]: -0.0511	BPF2[5]: 0.0487	BPF2[6]: 0.0918	BPF2[7]: 0.0426	BPF2[8]: -0.0391
BPF2[9]: -0.0705	BPF2[10]: -0.0312	BPF2[11]: 0.0271	BPF2[12]: 0.0457	BPF2[13]: 0.0186
BPF2[14]: -0.0145	BPF2[15]: -0.0210	BPF2[16]: -0.0067	BPF2[17]: 0.0032	BPF2[18]: 0.0000
BPF2[19]: -0.0029	BPF2[20]: 0.0054	BPF2[21]: 0.0149	BPF2[22]: 0.0091	BPF2[23]: -0.0104
BPF2[24]: -0.0224	BPF2[25]: -0.0117	BPF2[26]: 0.0117	BPF2[27]: 0.0229	BPF2[28]: 0.0108
BPF2[29]: -0.0100	BPF2[30]: -0.0177	BPF2[31]: -0.0076	BPF2[32]: 0.0061	BPF2[33]: 0.0092
BPF2[34]: 0.0031	BPF2[35]: -0.0015	BPF2[36]: -0.0000	BPF2[37]: 0.0014	BPF2[38]: -0.0027
BPF2[39]: -0.0077	BPF2[40]: -0.0048	BPF2[41]: 0.0056	BPF2[42]: 0.0122	BPF2[43]: 0.0064
BPF2[44]: -0.0066	BPF2[45]: -0.0130	BPF2[46]: -0.0062	BPF2[47]: 0.0058	BPF2[48]: 0.0104
BPF2[49]: 0.0045	BPF2[50]: -0.0037	BPF2[51]: -0.0056	BPF2[52]: -0.0019	BPF2[53]: 0.0009
BPF2[54]: 0.0000	BPF2[55]: -0.0009	BPF2[56]: 0.0017	BPF2[57]: 0.0049	BPF2[58]: 0.0031
BPF2[59]: -0.0036	BPF2[60]: -0.0079	BPF2[61]: -0.0042	BPF2[62]: 0.0043	BPF2[63]: 0.0086
BPF2[64]: 0.0041	BPF2[65]: -0.0039	BPF2[66]: -0.0070		

3.6 | Designing and Implementation of Final Multi-BandPass Filter

This was the only line for the multibandpass filter implementation

```
final_filter = bpf1 + bpf2
```

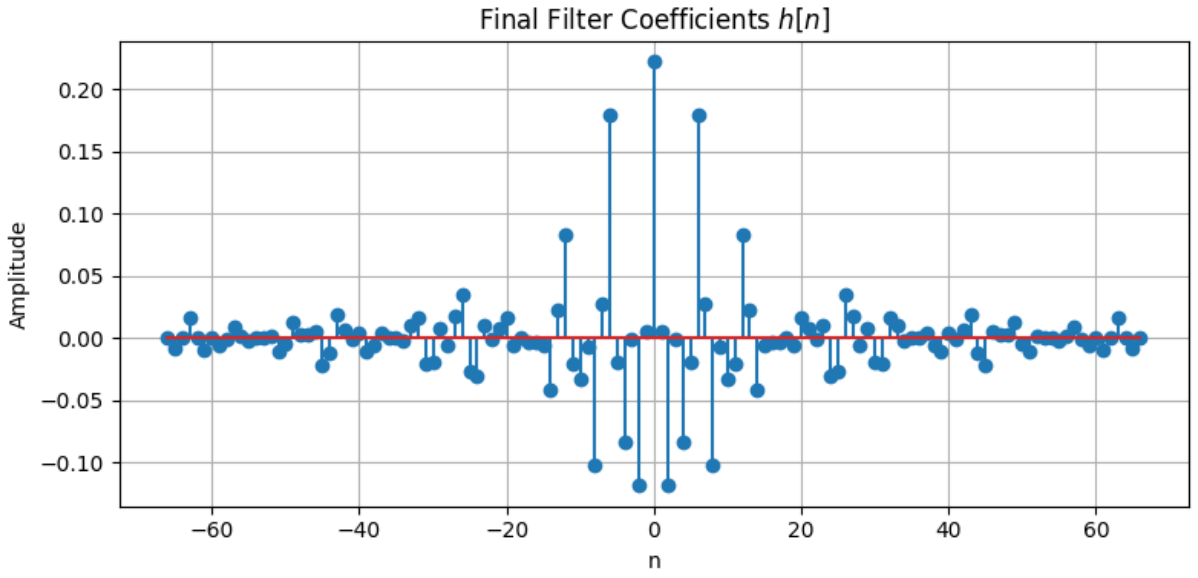


Figure 3.22: Final Filter Coefficients

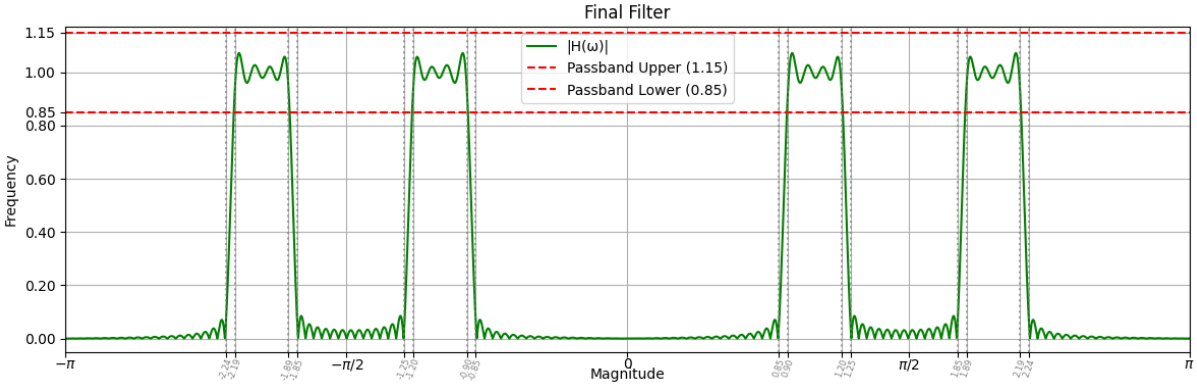


Figure 3.23: Final Filter Magnitude Response

Table 3.5: Final Filter 2 Coefficients

final_filter[-66]: -0.0001	final_filter[-65]: -0.0084	final_filter[-64]: 0.0004	final_filter[-63]: 0.0171
final_filter[-62]: 0.0004	final_filter[-61]: -0.0091	final_filter[-60]: -0.0001	final_filter[-59]: -0.0059
final_filter[-58]: -0.0012	final_filter[-57]: 0.0095	final_filter[-56]: 0.0011	final_filter[-55]: -0.0023
final_filter[-54]: -0.0000	final_filter[-53]: 0.0010	final_filter[-52]: 0.0014	final_filter[-51]: -0.0102
final_filter[-50]: -0.0046	final_filter[-49]: 0.0130	final_filter[-48]: 0.0028	final_filter[-47]: 0.0027
final_filter[-46]: 0.0060	final_filter[-45]: -0.0211	final_filter[-44]: -0.0120	final_filter[-43]: 0.0193
final_filter[-42]: 0.0061	final_filter[-41]: -0.0005	final_filter[-40]: 0.0047	final_filter[-39]: -0.0105
final_filter[-38]: -0.0063	final_filter[-37]: 0.0042	final_filter[-36]: 0.0000	final_filter[-35]: 0.0008
final_filter[-34]: -0.0026	final_filter[-33]: 0.0099	final_filter[-32]: 0.0166	final_filter[-31]: -0.0205
final_filter[-30]: -0.0190	final_filter[-29]: 0.0084	final_filter[-28]: -0.0058	final_filter[-27]: 0.0178
final_filter[-26]: 0.0345	final_filter[-25]: -0.0271	final_filter[-24]: -0.0306	final_filter[-23]: 0.0103
final_filter[-22]: -0.0008	final_filter[-21]: 0.0074	final_filter[-20]: 0.0161	final_filter[-19]: -0.0052
final_filter[-18]: 0.0000	final_filter[-17]: -0.0031	final_filter[-16]: -0.0031	final_filter[-15]: -0.0056
final_filter[-14]: -0.0418	final_filter[-13]: 0.0233	final_filter[-12]: 0.0834	final_filter[-11]: -0.0205
final_filter[-10]: -0.0328	final_filter[-9]: -0.0070	final_filter[-8]: -0.1014	final_filter[-7]: 0.0278
final_filter[-6]: 0.1794	final_filter[-5]: -0.0193	final_filter[-4]: -0.0837	final_filter[-3]: -0.0012
final_filter[-2]: -0.1180	final_filter[-1]: 0.0048	final_filter[0]: 0.2222	final_filter[1]: 0.0048
final_filter[2]: -0.1180	final_filter[3]: -0.0012	final_filter[4]: -0.0837	final_filter[5]: -0.0193
final_filter[6]: 0.1794	final_filter[7]: 0.0278	final_filter[8]: -0.1014	final_filter[9]: -0.0070
final_filter[10]: -0.0328	final_filter[11]: -0.0205	final_filter[12]: 0.0834	final_filter[13]: 0.0233
final_filter[14]: -0.0418	final_filter[15]: -0.0056	final_filter[16]: -0.0031	final_filter[17]: -0.0031
final_filter[18]: 0.0000	final_filter[19]: -0.0052	final_filter[20]: 0.0161	final_filter[21]: 0.0074
final_filter[22]: -0.0008	final_filter[23]: 0.0103	final_filter[24]: -0.0306	final_filter[25]: -0.0271
final_filter[26]: 0.0345	final_filter[27]: 0.0178	final_filter[28]: -0.0058	final_filter[29]: 0.0084
final_filter[30]: -0.0190	final_filter[31]: -0.0205	final_filter[32]: 0.0166	final_filter[33]: 0.0099
final_filter[34]: -0.0026	final_filter[35]: 0.0008	final_filter[36]: 0.0000	final_filter[37]: 0.0042
final_filter[38]: -0.0063	final_filter[39]: -0.0105	final_filter[40]: 0.0047	final_filter[41]: -0.0005
final_filter[42]: 0.0061	final_filter[43]: 0.0193	final_filter[44]: -0.0120	final_filter[45]: -0.0211
final_filter[46]: 0.0060	final_filter[47]: 0.0027	final_filter[48]: 0.0028	final_filter[49]: 0.0130
final_filter[50]: -0.0046	final_filter[51]: -0.0102	final_filter[52]: 0.0014	final_filter[53]: 0.0010
final_filter[54]: -0.0000	final_filter[55]: -0.0023	final_filter[56]: 0.0011	final_filter[57]: 0.0095
final_filter[58]: -0.0012	final_filter[59]: -0.0059	final_filter[60]: -0.0001	final_filter[61]: -0.0091
final_filter[62]: 0.0004	final_filter[63]: 0.0171	final_filter[64]: 0.0004	final_filter[65]: -0.0084
final_filter[66]: -0.0001			

Very clearly, the magnitude response of the filter is within the given tolerance limits and meets our requirement specifications.
Let us see how the phase plot of this filter looks:

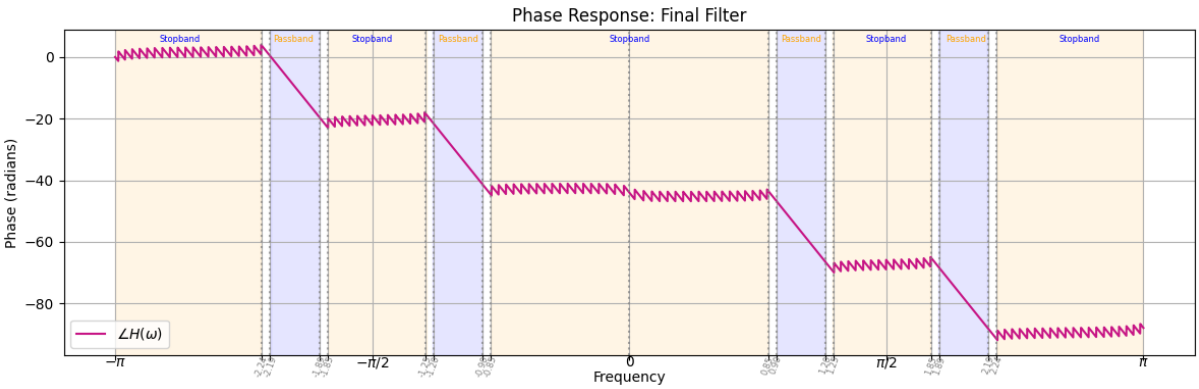


Figure 3.24: Phase Response of the Final Filter

- We see that in the stopband region, the phase is almost a constant (very slightly varying or ripllying around a constant, less variance)
- In the passband region, the phase response is linear.
- This is the pseudo linear response.

4 | Cool Codes

These are some of the code snippets written by me which I find cool.

4.1 | Universal Magnitude response Plotting Code

```
    def plot_manual_dtft(h,k1,k2,band_list,title,N,ylim=[-0.05,1.17],delta=0.075):
    omega, H = compute_dtft(h,N)
    plt.figure(figsize=(12, 4))
    plt.plot(omega, np.abs(H), color='green', label='|H(  )|')
    if(k1==1):
        upper = 1+delta
        lower = 1- delta
        plt.axhline(1 + delta, color='red', linestyle='--', label=f'Passband Upper ({1 +
    delta})')
        plt.axhline(1 - delta, color='red', linestyle='--', label=f'Passband Lower ({1 -
    delta})')
        yticks = plt.yticks()[0].tolist()
        yticks.extend([upper, lower])
        plt.yticks(sorted(set(yticks)))
    if(k2 == 1):
        band_edges = [w3[i] for i in band_list]
        for freq in band_edges:
         plt.axvline(freq, color='gray', linestyle=':', alpha=0.7)
         extra_xticks = [w3[i] for i in band_list]
    else: extra_xticks = []
    plt.title(title)
    plt.xlabel("Magnitude")
    plt.ylabel("Frequency")
    plt.xlim([-np.pi, np.pi])
    plt.ylim(ylim)
    default_xticks = [-np.pi, -np.pi/2, 0, np.pi/2, np.pi]
    default_labels = [r'$-\pi$', r'$-\pi/2$', '0', r'$\pi/2$', r'$\pi$']
    all_xticks = sorted(set(default_xticks + extra_xticks))
    ax = plt.gca()
    ax.set_xticks(all_xticks)
    ax.set_xticklabels(['' for _ in all_xticks])  # hide them
    for tick in all_xticks:
     if tick in default_xticks:
        idx = default_xticks.index(tick)
        label = default_labels[idx]
        ax.text(tick, ax.get_ylim()[0] - 0.02, label, ha='center', va='top', fontsize=10,
     usetex=False)
```

```
     else:
        ax.text(tick, ax.get_ylim()[0] - 0.02, f'{tick:.2f}', ha='center', va='top',
    fontsize=6, color='gray', rotation =70)
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

4.2 | Tolerance Checker

This code takes in the filter coefficients, the passband and the stopband pairs, N and the tolerance as the input and checks if there is a violation of the tolerance limit in each of the bands. I did this check for each and every filter i designed in the process: The low pass filters, the bandpass filters and the final filter as well.

```
def tolerance_checker(filter_coeffs,passband_pairs, stopband_pairs,N,delta=0.075):
 passband_tol = (1-delta,1+delta)
 stopband_tol = delta
 omega, H1 = compute_dtft(filter_coeffs,N)
 mag = np.abs(H1)
 for left, right in passband_pairs:
    indices = np.where((omega >= left) & (omega <= right))[0]
    band_mag = mag[indices]
    band_min = np.min(band_mag)
    band_max = np.max(band_mag)
    print(f"Passband [{left:.3f}, {right:.3f}]: min = {band_min:.6f}, max = {band_max:.6f
    }")
    if band_min < passband_tol[0] or band_max > passband_tol[1]:
        print("X Passband out of tolerance!\n")
    else:
        print("Good ! Passband within tolerance.\n")

 for left, right in stopband_pairs:
    indices = np.where((omega >= left) & (omega <= right))[0]
    band_mag = mag[indices]
    band_max = np.max(band_mag)
    print(f"Stopband [{left:.3f}, {right:.3f}]: max = {band_max:.6f}")
    if band_max > stopband_tol:
        print("X Stopband out of tolerance!\n")
    else:
        print("Good! Stopband within tolerance.\n")
```

For the Final Filter, i got an output of the form:

```
Passband [-2.194, -1.895]: min = 0.938607, max = 1.072120
✅ Passband within tolerance.

Passband [-1.197, -0.898]: min = 0.937053, max = 1.072016
✅ Passband within tolerance.

Passband [0.898, 1.197]: min = 0.937053, max = 1.072016
✅ Passband within tolerance.

Passband [1.895, 2.194]: min = 0.938607, max = 1.072120
✅ Passband within tolerance.

Stopband [-3.142, -2.244]: max = 0.070647
✅ Stopband within tolerance.

Stopband [-1.845, -1.247]: max = 0.085216
✅ Stopband within tolerance.

Stopband [-0.848, 0.848]: max = 0.070761
✅ Stopband within tolerance.

Stopband [1.247, 1.845]: max = 0.085216
✅ Stopband within tolerance.

Stopband [2.244, 3.142]: max = 0.070647
✅ Stopband within tolerance.
```

Figure 4.1: Output of the Tolerance checker over the Final Filter

4.3 | Transfer Function

$$
\begin{aligned}
H(z) = 0.00654441789021803 &- \frac{0.022967682402231}{z} - \frac{0.0129847660571661}{z^2} + \frac{0.0208510862870177}{z^3} + \frac{0.00656343923121179}{z^4} \\
&- \frac{0.000506071245735948}{z^5} + \frac{0.00506430469995528}{z^6} - \frac{0.0111436293863623}{z^7} - \frac{0.00665636158881766}{z^8} + \frac{0.00438927986679857}{z^9} \\
&+ \frac{2.51589970554731 \cdot 10^{-17}}{z^{10}} + \frac{0.000840304135424273}{z^{11}} - \frac{0.00269914340174221}{z^{12}} + \frac{0.0103665822353876}{z^{13}} + \frac{0.0173056851041171}{z^{14}} \\
&- \frac{0.021289406343366}{z^{15}} - \frac{0.0197509918457854}{z^{16}} + \frac{0.00869442788404572}{z^{17}} - \frac{0.00595700195772633}{z^{18}} + \frac{0.0183317980016775}{z^{19}} \\
&+ \frac{0.0354244305793154}{z^{20}} - \frac{0.0277980688047231}{z^{21}} - \frac{0.0313646649515825}{z^{22}} + \frac{0.0104963536002844}{z^{23}} - \frac{0.000791382364613808}{z^{24}} \\
&+ \frac{0.00757880681389978}{z^{25}} + \frac{0.0163167257352698}{z^{26}} - \frac{0.00530214958625779}{z^{27}} + \frac{8.75980288830194 \cdot 10^{-17}}{z^{28}} - \frac{0.00312247309208971}{z^{29}} \\
&- \frac{0.00311852874371751}{z^{30}} - \frac{0.00566481521965488}{z^{31}} - \frac{0.0420813208203662}{z^{32}} + \frac{0.0234214875299496}{z^{33}} + \frac{0.0839048505183404}{z^{34}} \\
&- \frac{0.0205693018395925}{z^{35}} - \frac{0.0329104293280675}{z^{36}} - \frac{0.00700501964147401}{z^{37}} - \frac{0.101654343087544}{z^{38}} + \frac{0.0278903485353427}{z^{39}} \\
&+ \frac{0.179693965821828}{z^{40}} - \frac{0.0193351881878003}{z^{41}} - \frac{0.0837334969055346}{z^{42}} - \frac{0.00118508614434878}{z^{43}} - \frac{0.117985431905847}{z^{44}} \\
&+ \frac{0.00484081826659946}{z^{45}} + \frac{0.222222222222222}{z^{46}} + \frac{0.00484081826659946}{z^{47}} - \frac{0.117985431905847}{z^{48}} - \frac{0.00118508614434878}{z^{49}} \\
&- \frac{0.0837334969055346}{z^{50}} - \frac{0.0193351881878003}{z^{51}} + \frac{0.179693965821828}{z^{52}} + \frac{0.0278903485353427}{z^{53}} - \frac{0.101654343087544}{z^{54}} \\
&- \frac{0.00700501964147401}{z^{55}} - \frac{0.0329104293280675}{z^{56}} - \frac{0.0205693018395925}{z^{57}} + \frac{0.0839048505183404}{z^{58}} + \frac{0.0234214875299496}{z^{59}} \\
&- \frac{0.0420813208203662}{z^{60}} - \frac{0.00566481521965488}{z^{61}} - \frac{0.00311852874371751}{z^{62}} - \frac{0.00312247309208971}{z^{63}} + \frac{8.75980288830194 \cdot 10^{-17}}{z^{64}} \\
&- \frac{0.00530214958625779}{z^{65}} + \frac{0.0163167257352698}{z^{66}} + \frac{0.00757880681389978}{z^{67}} - \frac{0.000791382364613808}{z^{68}} + \frac{0.0104963536002844}{z^{69}} \\
&- \frac{0.0313646649515825}{z^{70}} - \frac{0.0277980688047231}{z^{71}} + \frac{0.0354244305793154}{z^{72}} + \frac{0.0183317980016775}{z^{73}} - \frac{0.00595700195772633}{z^{74}} \\
&+ \frac{0.00869442788404572}{z^{75}} - \frac{0.0197509918457854}{z^{76}} - \frac{0.021289406343366}{z^{77}} + \frac{0.0173056851041171}{z^{78}} + \frac{0.0103665822353876}{z^{79}} \\
&- \frac{0.00269914340174221}{z^{80}} + \frac{0.000840304135424273}{z^{81}} + \frac{2.51589970554731 \cdot 10^{-17}}{z^{82}} + \frac{0.00438927986679857}{z^{83}} - \frac{0.00665636158881766}{z^{84}} \\
&- \frac{0.0111436293863623}{z^{85}} + \frac{0.00506430469995528}{z^{86}} - \frac{0.000506071245735948}{z^{87}} + \frac{0.00656343923121179}{z^{88}} + \frac{0.0208510862870177}{z^{89}} \\
&- \frac{0.0129847660571661}{z^{90}} - \frac{0.022967682402231}{z^{91}} + \frac{0.00654441789021803}{z^{92}}
\end{aligned}
$$

5 | Optimization for Better Economy

The code for iteration 1 ran perfectly, but given the constraints, it overperfomed (i.e. its ripples arent even close to 1 ± 0.15), which could probably because of overuse of resources. There could be a possibility of optimizing the values used and overlooking certain steps, to get a multi-bandpass filter which still satisfies the tolerance conditions but could be obtained with a lesser N value or lesser alpha value. This is the next stage of checking that out (iteration 2). Last time, i made sure each low pass filter satisfied the tolerance of 0.075, but i will let this constraint loose, and see if still the resulting multibandpass filter would satisfy the tolerance conditions. This time I will let the intermediate filters not satisfy the tolerance limits as long as the final filter is respecting the tolerance limits.

5.1 | Procedure

I let the tolerance for each individual bandpass filter be 0.15 and calculated the kaiser window parameters for these. After fine tuning i got the following values:

1. The value of delta_w_t is, 0.049866550056980596
2. The value of A is, 16.478174818886377
3. The minimum value of N is : 38
4. We use the value of N to be : 46

5. Alpha value given by the equation is: 0

6. The tweaked value of alpha is 0

Clearly the value of N which was 66 in the first iteration has reduced to 46 in the second iteration and the alpha value has also reduced to 0, therefore this is a lot of improvement for the given conditions.

5.2 | The Changes

Let the low pass filter not respect tolerance limits but the band pass filter 1 still does. The end final filter still respects all the limits.

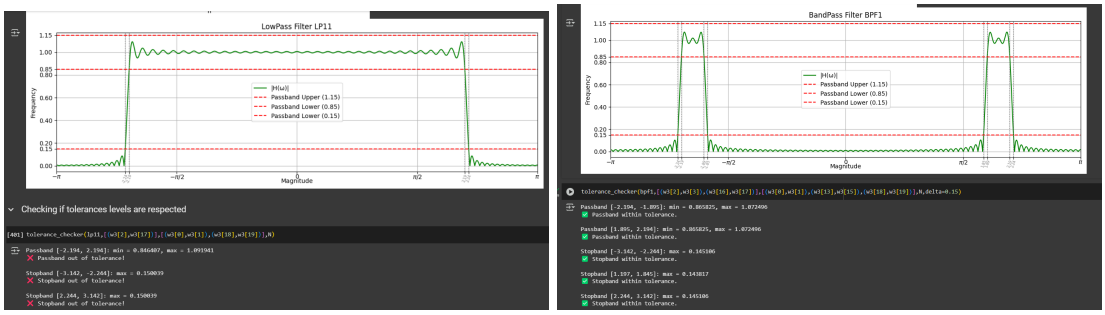


Figure 5.1: Low Pass not respecting tolerance limits but Bandpass filter still respects limits

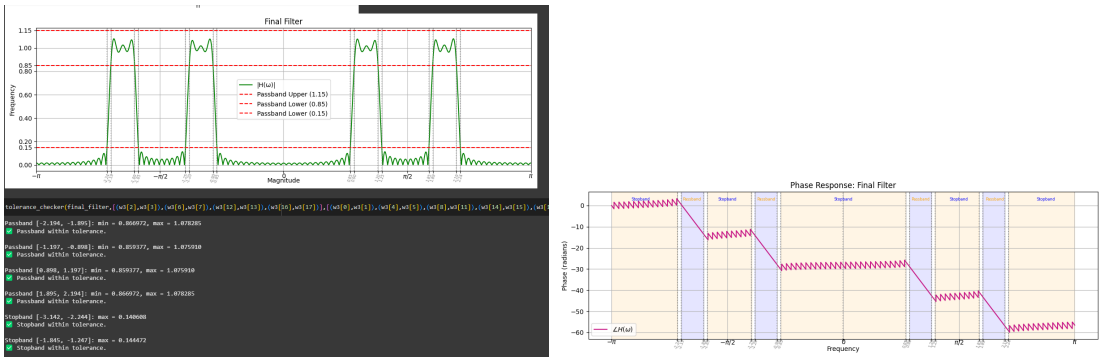


Figure 5.2: Final filter still repects all the tolerance limits and works perfectly and the phase plot

5.3 | Transfer Function

$$
\begin{aligned}
H(z) = & -7.77014566818567 \cdot 10^{-5} - \frac{0.00835442995448044}{z} + \frac{0.000361334124791582}{z^2} + \frac{0.0171184751499612}{z^3} + \frac{0.00037710824925036}{z^4} \\
& - \frac{0.00909994144095782}{z^5} - \frac{8.83346340730777 \cdot 10^{-5}}{z^6} - \frac{0.00585295731589764}{z^7} - \frac{0.00121611273600442}{z^8} + \frac{0.00954255516400874}{z^9} \\
& + \frac{0.00111416211089544}{z^{10}} - \frac{0.00230012601638868}{z^{11}} - \frac{1.00244415208159 \cdot 10^{-16}}{z^{12}} + \frac{0.000974884338999428}{z^{13}} + \frac{0.00141640653029426}{z^{14}} \\
& - \frac{0.0102387328571992}{z^{15}} - \frac{0.00460961739945026}{z^{16}} + \frac{0.0129803283458709}{z^{17}} + \frac{0.00278899945404342}{z^{18}} + \frac{0.00266374224083603}{z^{19}} \\
& + \frac{0.00600000505865842}{z^{20}} - \frac{0.0211374522559137}{z^{21}} - \frac{0.011994582950945}{z^{22}} + \frac{0.0193310818806904}{z^{23}} + \frac{0.00610655810730485}{z^{24}} \\
& - \frac{0.00047247140990591}{z^{25}} + \frac{0.0047439929246099}{z^{26}} - \frac{0.0104730426097907}{z^{27}} - \frac{0.00627576934603174}{z^{28}} + \frac{0.00415115761350036}{z^{29}} \\
& + \frac{2.36468939350743 \cdot 10^{-17}}{z^{30}} + \frac{0.000799449358781834}{z^{31}} - \frac{0.00257521421289473}{z^{32}} + \frac{0.00991787448578924}{z^{33}} + \frac{0.0166008462389789}{z^{34}} \\
& - \frac{0.0204751151382786}{z^{35}} - \frac{0.0190430352032638}{z^{36}} + \frac{0.00840303066241083}{z^{37}} - \frac{0.00577076919298046}{z^{38}} + \frac{0.0177985865271683}{z^{39}} \\
& + \frac{0.0344684219617843}{z^{40}} - \frac{0.027104095214932}{z^{41}} - \frac{0.0306426632285873}{z^{42}} + \frac{0.0102743363330824}{z^{43}} - \frac{0.000776059759088208}{z^{44}} \\
& + \frac{0.0074450427927685}{z^{45}} + \frac{0.0160553997307131}{z^{46}} - \frac{0.00522547891370351}{z^{47}} + \frac{8.75417941309927 \cdot 10^{-17}}{z^{48}} - \frac{0.00308629881999264}{z^{49}} \\
& - \frac{0.00308651432212843}{z^{50}} - \frac{0.00561368617945442}{z^{51}} - \frac{0.041750358173677}{z^{52}} + \frac{0.0232626110380381}{z^{53}} + \frac{0.0834197589980744}{z^{54}} \\
& - \frac{0.0204693512296372}{z^{55}} - \frac{0.0327782354860762}{z^{56}} - \frac{0.00698222357321748}{z^{57}} - \frac{0.101392916275768}{z^{58}} + \frac{0.0278354243499246}{z^{59}} \\
& + \frac{0.179433944037929}{z^{60}} - \frac{0.0193157563876677}{z^{61}} - \frac{0.0836796345346703}{z^{62}} - \frac{0.00118465730851869}{z^{63}} - \frac{0.117966455666879}{z^{64}} \\
& + \frac{0.00484062361659485}{z^{65}} + \frac{0.222222222222222}{z^{66}} + \frac{0.00484062361659485}{z^{67}} - \frac{0.117966455666879}{z^{68}} - \frac{0.00118465730851869}{z^{69}} \\
& - \frac{0.0836796345346703}{z^{70}} - \frac{0.0193157563876677}{z^{71}} + \frac{0.179433944037929}{z^{72}} + \frac{0.0278354243499246}{z^{73}} - \frac{0.101392916275768}{z^{74}} \\
& - \frac{0.00698222357321748}{z^{75}} - \frac{0.0327782354860762}{z^{76}} - \frac{0.0204693512296372}{z^{77}} + \frac{0.0834197589980744}{z^{78}} + \frac{0.0232626110380381}{z^{79}} \\
& - \frac{0.041750358173677}{z^{80}} - \frac{0.00561368617945442}{z^{81}} - \frac{0.00308651432212843}{z^{82}} - \frac{0.00308629881999264}{z^{83}} + \frac{8.75417941309927 \cdot 10^{-17}}{z^{84}} \\
& - \frac{0.00522547891370351}{z^{85}} + \frac{0.0160553997307131}{z^{86}} + \frac{0.0074450427927685}{z^{87}} - \frac{0.000776059759088208}{z^{88}} + \frac{0.0102743363330824}{z^{89}} \\
& - \frac{0.0306426632285873}{z^{90}} - \frac{0.027104095214932}{z^{91}} + \frac{0.0344684219617843}{z^{92}} + \frac{0.0177985865271683}{z^{93}} - \frac{0.00577076919298046}{z^{94}} \\
& + \frac{0.00840303066241083}{z^{95}} - \frac{0.0190430352032638}{z^{96}} - \frac{0.0204751151382786}{z^{97}} + \frac{0.0166008462389789}{z^{98}} + \frac{0.00991787448578924}{z^{99}} \\
& - \frac{0.00257521421289473}{z^{100}} + \frac{0.000799449358781834}{z^{101}} + \frac{2.36468939350743 \cdot 10^{-17}}{z^{102}} + \frac{0.00415115761350036}{z^{103}} - \frac{0.00627576934603174}{z^{104}} \\
& - \frac{0.0104730426097907}{z^{105}} + \frac{0.0047439929246099}{z^{106}} - \frac{0.00047247140990591}{z^{107}} + \frac{0.00610655810730485}{z^{108}} + \frac{0.0193310818806904}{z^{109}} \\
& - \frac{0.011994582950945}{z^{110}} - \frac{0.0211374522559137}{z^{111}} + \frac{0.00600000505865842}{z^{112}} + \frac{0.00266374224083603}{z^{113}} + \frac{0.00278899945404342}{z^{114}} \\
& + \frac{0.0129803283458709}{z^{115}} - \frac{0.00460961739945026}{z^{116}} - \frac{0.0102387328571992}{z^{117}} + \frac{0.00141640653029426}{z^{118}} + \frac{0.000974884338999428}{z^{119}} \\
& - \frac{1.00244415208159 \cdot 10^{-16}}{z^{120}} - \frac{0.00230012601638868}{z^{121}} + \frac{0.00111416211089544}{z^{122}} + \frac{0.00954255516400874}{z^{123}} - \frac{0.00121611273600442}{z^{124}} \\
& - \frac{0.00585295731589764}{z^{125}} - \frac{8.83346340730777 \cdot 10^{-5}}{z^{126}} - \frac{0.00909994144095782}{z^{127}} + \frac{0.00037710824925036}{z^{128}} + \frac{0.0171184751499612}{z^{129}} \\
& + \frac{0.000361334124791582}{z^{130}} - \frac{0.00835442995448044}{z^{131}} - \frac{7.77014566818567 \cdot 10^{-5}}{z^{132}}
\end{aligned}
$$

6 | Reviews and Reviewers

1. Reviewed by Aman and Sarvadnya
2. Aman Rishal C H (22B3914): A well-structured and technically sound report that effectively applies the Kaiser window method for FIR multi-bandpass filter design. The design meets all specifications, and the methodology is clearly explained, though a few formatting improvements and more emphasis on final response plots would enhance it further.

3. Sarvadnya Purkar (22B4232): The report thoroughly details the design of a multi-bandpass FIR filter using a rectangular Kaiser window, with clear explanations of the steps, from calculations to implementation. The filter's performance meets all the required specifications, and the comparison with IIR filters highlights the trade-offs well. Some minor formatting adjustments could improve readability, but the overall analysis is solid.